UNC

SECURI

(2)

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY: UNCLASSIFIED	3. DISTRIBUTION/AVAILABILITY OF REPORT: APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR- 93 0141
6a. NAME OF PERFORMING ORGANIZATION: Oregon State University; 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION: AFOSR/NM
6c. ADDRESS (City, State and ZIP Code): Computer Science Dept. Corvallis, OR 97331-3202	7b. ADDRESS (City, State and ZIP Code): Bolling AFB DC 20332-6448 110 Duncan Ave/Suite B115 Bolling AFB DC 20332-0001
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR; 8b. OFFICE SYMBOL (If applicable): NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR-90-0348
8c. ADDRESS (City, State and ZIP Code): Bolling AFB DC 20332-6448	10. SOURCE OF FUNDING NOS. — PROGRAM ELEMENT NO.: 61102F; PROJECT NO.: 2304; TASK NO.; WORK UNIT NO.
11. TITLE (Include Security Classification): Real-Time Value Driven Monitoring and Repair (U)	
12. PERSONAL AUTHOR(S): Bruce D'Ambrosio	
13a. TYPE OF REPORT: Final; 13b. TIME COVERED: FROM 9/1/90 TO 8/31/92	14. DATE OF REPORT (Yr., Mo., Day): 12/30/92; 15. PAGE COUNT: 38
16. SUPPLEMENTARY NOTATION	
17. COSATI CODES — GROUP; SUB. GR.	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number): Intelligent Real-Time Problem Solving

TRACT

Monitoring and repair (diagnosis, for short) are often thought of as isolated tasks in theoretical reasoning (reasoning with the goal of updating our beliefs about the world). We present a decision-theoretic interpretation of diagnosis as a task in practical reasoning (reasoning with the goal of acting in the world), and sketch components of our approach to this task. These components include an abstract problem description, a decision-theoretic model of the basic task, a set of inference methods suitable for evaluating the decision representation in real-time, and a control architecture to provide the needed continuing coordination between the agent and its environment. A principal contribution of this work is the representation and inference methods we have developed, which extend previously available probabilistic inference methods and narrow, somewhat, the gap between probabilistic and logical models of diagnosis.

TRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED XX SAME AS RPT. XX DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: Dr Abraham Waksman	22b. TELEPHONE NUMBER (Include Area Code): 202/767-5028; 22c. OFFICE SYMBOL: NM

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE. — UNCLASSIFIED SECURITY CLASSIFICATION OF THIS PAGE

Report OSU CS 92-30-08

Real-Time Value-Driven Monitoring and Repair

Bruce D'Ambrosio
Computer Science Dept.
Oregon State University
Corvallis, OR 97331-3202

30 December 1992

Final Report, AFOSR 90-0348

Unclassified

Prepared for

Air Force Office of Scientific Reearch
Bolling AFB, DC 20332-6448

Contents

1 Introduction 1
- 1.1 On-Line Equipment Maintenance 1

2 A decision-theoretic formulation of an OLMA 3

3 Incremental Probabilistic Inference 7
- 3.1 Representation 7
- 3.2 Inference 8
- 3.3 Term Computation 9
  - 3.3.1 Desiderata for a TCS 10
  - 3.3.2 Outline of Term Computation 12
- 3.4 Term Computation and Real-time Diagnosis 17

4 Reaction and deliberation in real-time diagnosis 19
- 4.1 Overview 20
- 4.2 Decision-making processes 20
  - 4.2.1 Models of decision processes 21
  - 4.2.2 Decision domains 21
- 4.3 Decision Situations 22
  - 4.3.1 Real-Time Decision Making 23
- 4.4 Reaction and Deliberation in the OLMA 25
- 4.5 Discussion 25
- 4.6 Summary 28

5 Summary 29

6 Grant Administration Data 30
- 6.1 Publication 30
- 6.2 Professionals involved 30
- 6.3 Interactions 30

Abstract

Monitoring and repair (diagnosis, for short) are often thought of as isolated tasks in theoretical reasoning (reasoning with the goal of updating our beliefs about the world). We present a decision-theoretic interpretation of diagnosis as a task in practical reasoning (reasoning with the goal of acting in the world), and sketch components of our approach to this task. These components include an abstract problem description, a decision-theoretic model of the basic task, a set of inference methods suitable for evaluating the decision representation in real-time, and a control architecture to provide the needed continuing coordination between the agent and its environment. A principal contribution of this work is the representation and inference methods we have developed, which extend previously available probabilistic inference methods and narrow, somewhat, the gap between probabilistic and logical models of diagnosis.

Chapter 1

Introduction

In this paper we will present a status report on research into diagnosis as an embedded value-driven activity. Several characteristics become apparent from this perspective that are not normally emphasized in work on diagnosis. First, we view the task as on-going rather than one-shot. Second, we recognize that diagnosis must often be performed under time limitations. Finally, we consider a decision-theoretic model of diagnostic activity in which diagnostic reasoning is not a theoretical activity but a practical one. That is, its goal is not to increase our knowledge, but rather to choose an action which maximizes expected utility.

Four key elements characterize our current approach to these problems. First, we have developed a simple abstract domain which captures these essential elements, which we title the On-Line Maintenance Agent (OLMA) task domain. Second, we have developed a decision-theoretic model of the basic decision task facing an agent in this domain. Third, we have developed probabilistic inference methods suited for solving the resulting decision problem. Fourth, we have developed a problem-solving architecture intended to provide the reactivity needed in embedded, ongoing tasks. We view the third contribution, the development of suitable probabilistic inference mechanisms, as our most significant contribution to date. These methods have been described elsewhere. The goal of this paper is to place them in the context of the larger task of real-time value-driven diagnosis. We begin with a review of the basic task domain, then discuss each of the remaining three elements of our approach. We close with a discussion of related work, limitations, and future directions.

1.1 On-Line Equipment Maintenance

Early writings by at least some in AI (e.g., [40]) stress the necessity of recognizing both the limitations and embedded nature of realizable agents. Despite that, much work in AI has proceeded as if the demands placed by the environment are static. In a diagnostic task, for example, the system being diagnosed is often presumed to have broken before diagnosis starts, and to not undergo any further changes (other than

those initiated by the diagnostic agent) while diagnosis takes place (an exception is some medical monitoring and diagnosis systems). Further, the environment is presumed to be static enough to allow all solution-quality/time-to-solution tradeoffs to be made at design time (the standard "20 minute solution time" criterion for judging potential expert system applications is a prime example of this). These assumptions, together with assumptions that problem solving proceeds by inference with explicit, categorical representations, characterize much work in AI, whether it is on planning, diagnosis, natural language understanding, or image understanding (although less so in these latter areas).

Diagnosis, in accord with the above, is often formulated as a static, detached process, the goal of which is the assessment of the exact (or most probable) state of some external system. In contrast, we view diagnosis as a dynamic, practical activity by an agent engaged with a changing and uncertain world. We have focused our initial investigations of diagnosis on the task of diagnosing a simple digital system in situ. Our formulation of embedded diagnosis has the following characteristics:

- The equipment [1] under diagnosis continues to operate while being diagnosed.
- Multiple faults can occur (and can continue to occur after an initial fault is detected).
- Faults can be intermittent.
- There is a known fixed cost per unit time the system is malfunctioning.
- The agent senses equipment operation through a set of fixed sensors and one or more movable probes.
- Action alternatives include probing test points and replacing individual components. Each action has a corresponding cost.
- The agent can only perform one action at a time.
- The overall task is to minimize total cost over some extended time period during which several failures can be expected to occur.

We term this task the *On-Line Maintenance* task, and an agent intended for performing such a task an *On-Line Maintenance Agent*. An interesting aspect of this reformulation of the problem is that diagnosis is not a direct goal. A precise diagnosis is neither always obtainable nor necessary. Indeed, it is not even obvious a priori what elements of a diagnosis are even relevant to the decision at hand.

[1] We will use "system" to refer to our diagnostic system, and "equipment" to refer to the target physical system.

Chapter 2

A decision-theoretic formulation of an OLMA

Our first commitment is that the task is essentially a decision-theoretic one. That is, the essential task of the agent is to *act* in the face of limited information. In order to formulate this problem decision-theoretically, the agent must have knowledge of several parameters of the situation: It must know the cost of each type of replacement or probe act, the cost of system outage, and expected probabilities of component failures over the next decision cycle[1]. A naive attempt to formulate this task decision-theoretically, however, encounters three problems. First, a proper decision-theoretic consideration of this task would require looking ahead over all decisions over the entire operational life of the equipment in order to optimize the first decision. This is clearly computationally intractable. Second, even if the first problem can be solved, time is passing while the agent is computing the first action, and it is not clear how the agent should trade quality of a decision for timeliness of the solution in choosing actions. Finally, the agent must act repeatedly, yet each action is in a new context: not only must a new set of input data be considered, but also a new set of beliefs about system state, based on prior computation.

The infinite lookahead, or "small worlds" [37] problem has two subproblems, one for replacement actions and another for probe actions. We circumvent the first subproblem, that of infinite lookahead for replacement actions, as follows. For replacement actions it is possible, under assumptions about stability of the fault transition probabilities and fault costs, to derive off-line an optimal replacement policy conditioned on current beliefs about component states (this would take the form: "replace component x whenever its posterior probability of being faulted is greater than p_x," where p_x depends on parameters of the target system). We use an alternate, equivalent form of this result that requires less a-priori analysis, an assumption of policy

[1]These latter two costs will vary with agent processing capacity, since a slower agent will take longer to make a decision. This will increase the chance of a component failing during a single decision cycle, and increase the cost of a system outage over a decision cycle.

stability. This assumption is roughly as follows: If I choose not to replace a component now, then, all other things being equal (ie, no new unexpected sense data), I will make the same choice next time. Under this assumption, the temporal consequences of a decision extend, not for a single sense/act cycle, but several decision cycles into the future. We model the temporal extent of the outcome state for a decision as fixed time-period chosen to satisfy the following constraints:

$$t >> r/f$$
$$t << r/pf$$

where:

t is the outcome state duration (effectively, the multiplier for failure costs),

r is the cost of component replacement,

f is the cost of component failure for a unit clock time, and

p is the probability of component failure during a unit time interval.

The first constraint arises from the observation that, if the cost of replacement is greater than the cost of failure over the outcome period, then our limited look-ahead agent will never replace a component. The second constraint arises from a more subtle problem: even if we are certain the equipment is functioning correctly now, there is a non-zero probability it will be in failure mode in the outcome state. If the expected cost of that failure ($p*f*t$) is greater than component replacement cost (r), then the agent will always choose to replace, even though it believes the equipment is functioning perfectly.

We resolve the second subproblem, that of determining the expected value of probe actions, by using the standard decision-theoretic heuristic of one-step look-ahead. The result of these two techniques gives us an abstract decision-basis for the first decision as shown in figure 2.1, where the link from the state at time zero to the value node reflect the costs of operating in that state for one decision time, and the link from the state at time one to the value node reflect the cost of operating in that state for d time units.

A few notes about this representation are in order. $S0$, $S1$, and $S2$ represent the state of the equipment at times 0, 1, and 2 respectively. The actual state values are unknown to the agent. $O0$ and $O1$ represent the observational data available to the agent at the time decisions $D0$ and $D1$ are to be made. The solid arcs from $S0$ and $S1$ to $O0$ and $O1$ respectively reflect the fact that $O0$ and $O1$ are directly influenced by the (unobservable) system state. The dashed arcs from $O0$ to $D0$ and $O1$ to $D1$ represent the fact that the actual values of these variables will be available at the time the respective decisions will be made. FInally, V represents the value of possible

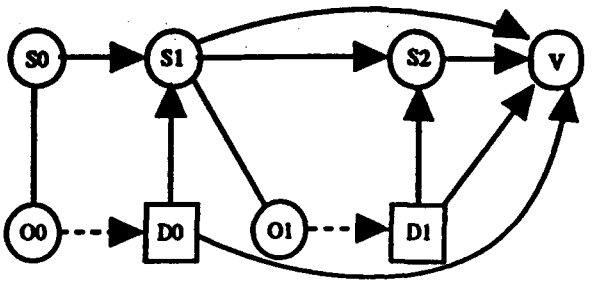


Figure 2.1: Abstract Decision Basis for first decision

outcomes, and the arcs to it reflect that the value is a function of the actual state of the equipment at times 0 and 1 as well as the direct costs of the actions the agent chooses.

We refer to this as an "abstract" decision basis because we typically will not represent the state of the system or the observational data as single state variables. Rather, we will typically have a structured representation, in the form of a belief net. $S0$, for example, represents the set of unobservable system state parameters and relationships among them, $O0$ represents the set of observed parameters, and the arc between $S0$ and $O0$ represents the relationship between the unobservable and observable system parameters. This general model can be instantiated with casual or evidential relationships (or both) between unobservable and observable parameters. Consider, for example, a simple sequential two inverter circuit. For that case, we might use causal knowledge to build a specific instantiation of our abstract $S0$ and $O0$ as shown in figure 2.2. In this figure, "Inv1, P, and Inv2" might all be components of "$S0$", and "I and O" might be directly observable.

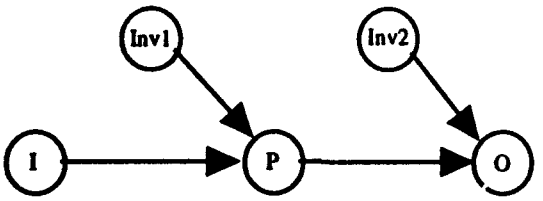


Figure 2.2: Decision Basis fragment for two inverter system

Our second problem was that of trading off quality of solution with time to solution. There are two issues here. First, if the equipment is faulted, the longer we delay taking a repair action, the higher the cost incurred. Second, since equipment operation is in parallel with agent operation, a fault may occur *while* the agent "wastes" time reasoning about a prior, correct set of sense data. We provide no solution to either of these problems here. However, we do present later an incremental (anytime) decision procedure, and sketch experiments we are performing to statistically infer

effective reasoning policies.

We resolve our final problem, that of making subsequent decisions, by simply extending the above decision basis forward in time by one decision each cycle. A sample decision basis for the second decision made by the maintenance agent is shown in figure 2.3. This method would seem to have a problem: one would expect that decision time (and space) would increase at least linearly with time. In fact, both time and space requirements are constant. We defer the discussion of how we accomplish this to the next section, on incremental probabilistic inference.

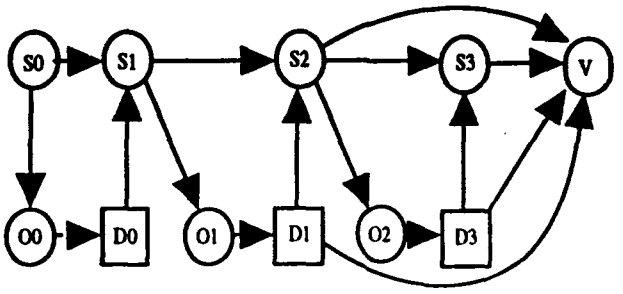


Figure 2.3: Abstract Decision Basis for second decision

In summary, then, our base-level agent executes the following cycle each time it is called upon to choose an action:

1. Extend the decision basis forward in time by one decision cycle.
2. Acquire current sense data (including probe values).
3. Find the action with minimum expected cost.
4. Post the selected act as evidence in the belief net, prune unneeded old cycles from the net, and return selected action.

Chapter 3

Incremental Probabilistic Inference

The problem of computing the expected utility for action alternatives can be cast as a belief-net inference problem, as shown by Cooper [7]. However, most current belief net inference facilities provide poor support for the cycle described above. The limitations in existing algorithms are both representational and inferential. The key representational limitation is an inability to represent structure within a single conditional probability distribution. The key inferential problems can be described as lack of *incrementality* with respect to various task requirements.

3.1 Representation

A belief net [32] is a compact, localized representation of a probabilistic model. The key to its locality is that, given a graphical structure representing the dependencies (and, implicitly, conditional independencies) among a set of variables, the joint probability distribution over that set can be completely described by specifying the appropriate set of marginal and conditional distributions over the variables involved. When the graph is sparse, this will involve a much smaller set of numbers than the full joint. Equally important, the graphical structure can be used to guide processing to find efficient ways to evaluate queries against the model. For more details, see [32], [9], [38], [29]. All is not as rosy at it might seem, though. The graphical level is not capable of representing all interesting structural information which might simplify representation or inference. The only mechanism available for describing antecedent interactions in typical general purpose belief net inference algorithms is the full conditional distribution across all antecedents. However, a number of restricted interaction models have been identified which have lower space and time complexity than the full conditional. The noisy-or [32], [33], [20], [21], [1] for example, can be used to model independent causes of an event, and inference complexity is linear in both space and time in the number of antecedents for many inferences. Similarly, various asymmetric

[39], [17] and logical relationships are inefficiently represented using a full conditional. Finally, value models used in utility modelling are often factored, for example they may be additive. We have developed an algebraic extension to belief nets which permits conditional distributions to be defined as algebraic compositions of smaller distributions [11]. We have shown that this representation is capable of capturing all known intra-distribution structures, and that simple inference algorithms can make use of this structure to perform inference effectively. Consider, for example, the problem of modelling the conditional distribution of the output of our two inverter circuit given the state of the second inverter and the output of the first inverter. This would normally be modeled as a single monolithic conditional distribution containing (if the inverter has four states: Ok, Stuck-at 0, Stuck-at 1, and Unknown) 16 numbers. The fact that three of the four inverter states are deterministic is lost. However, we model this as follows using our local expression language:

$$\begin{aligned} exp(O) \;=\; & P(O_1|Inv2 = Ok, I1out_0) \\ & +P(O_0|Inv2 = Ok, I1out_1) \\ & +P(O_0|Inv2 = S0) + P(O_1|Inv2 = S1) \\ & +P(O|Inv2 = Uk) \end{aligned}$$

A total of only six numbers are needed for the above (4 of them are 1.0). But more significantly, we have captured explicitly important structural information such as the fact that the output is independent of the input when the inverter is Stuck-at 1, Stuck-at 0, or Unknown, and that inverter behavior is deterministic in the Ok, Stuck-at 0, and Stuck-at 1 states.

We believe this representation goes a long way towards closing the expressiveness gap between probabilistic and propositional representations for device models.

3.2 Inference

Now that we have an adequate abstract model for the basic on-line maintenance decision and a representation adequate for expressing device and utility models, we need inference methods suited for the task at hand. Despite the fact that current state-of-the-art algorithms exploit the independence information in a belief net to construct efficient computations for probabilistic inference [32], [27], [38], in practice computational cost still grows rapidly [28], limiting application of these techniques to belief nets with a few hundred variables. Also, the services offered by current probabilistic reasoning systems are not well matched to the needs of higher-level problem solvers. As we discussed in [8] and [10], problem solvers typically interleave model construction, revision, and evaluation. As a specific example of this, consider the requirements posed by our formulation of the OLMA task:

- We must extend a model forward in time without discarding all previous inference and starting over. Current methods lack the incrementality with respect

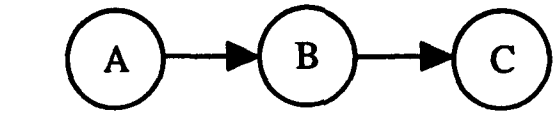
to model reformulation operations needed to support this.

- Our decision evaluation methods require that we query arbitrary subsets of the network variables. Current methods lack the incrementality with respect to queries needed to support this.
- The embedded nature of the task requires that we have flexible methods for performing partial inference. Current methods lack the incrementality with respect to completeness needed to support this.

In summary, no existing general-purpose low-level (propositional) probabilistic representation service provides incrementality with respect to model revision and resource usage in a theoretically sound manner. We, of course, have an answer. In this subsection we begin by reviewing the inference problem in a more general setting, and offering a redefinition of the basic inference task. We sketch how an inference engine which performs this task can serve as the core of an incremental probabilistic representation service, and report on the status of our current implementation.

3.3 Term Computation

Probabilistic inference in belief nets, as currently defined, is generally taken to be the computation of the prior or posterior marginal, conjunctive, or conditional probability distribution over some subset of variables in a fixed network. This is often an unnecessarily restrictive formulation of the problem. The actual computation of any prior or posterior probability can in general be viewed as a sum over a number of terms (in the extreme case, this occurs as marginalization of the full joint). While the number of terms to be computed is exponential in the size of the network, the time complexity of computation of a single term is linear. Consider the network shown in figure 3.1.



P(A)	
t	f
.95	.05

P(B)	t	f
At	.9	.1
Af	.15	.85

P(C)	t	f
Bt	.75	.25
Bf	.2	.8

Figure 3.1: Simple Belief Net

In this network:

$P(C_t) = \sum_{A,B} P(C_t|B,A) * P(B) * P(A)$
$= .95 * .9 * .75 + .95 * .1 * .2 + .05 * .85 * .2 + .05 * .15 * .75$
$= .64125 + .019 + .0085 + .005625$

We take the computation of a single term as an appropriate primitive task for probabilistic inference, and next show how an inference system with the needed incrementality properties can be built around it.

3.3.1 Desiderata for a TCS

We believe a low-level representation service should have two key properties: it should be *Incremental* and *Efficient.*

Definition 1 *A system is Incremental with respect to some capability to the extent that it can make use of the results of previous computations to reduce the cost or improve the quality of results for subsequent computations.*

For example, a system would be incremental with respect to queries if it took advantage of results computed during processing of earlier queries in the processing of some subsequent query. We identify four aspects of incrementality possible in probabilistic inference:

1. Resource incrementality: Computation of an exact posterior probability is an NP-hard task. Therefore, any practically usable system must offer facilities for computing approximate responses to queries. Incrementality with respect to resources enables a system to use increments of time to refine estimates. This give the problem solver control over the time/quality tradeoff in inference.

2. Query incrementality: Many probabilistic inference systems automatically compute the answer to a fixed set of queries (eg, the set of marginal probabilities for all the nodes in the net), and most have no capability to process any other queries. Incrementality with respect to queries enables a system to accept multiple queries, and to use partial results computed during processing of earlier queries to simplify processing of subsequent queries.

3. Evidence incrementality: Evidence typically arrives over time: A robot turns to scan a new part of the scene, a medical lab reports a new test result, and so on. Incrementality with respect to evidence enables a system to update its internal representations when new evidence arrives, rather than recompute all queries from the initial belief net. Most modern belief net algorithms possess evidence incrementality.

4. Representation incrementality: A belief net is an improverished form of representation: it is a minor extension of a propositional logic. And yet, inference within this representation is NP-hard. We believe, therefore, that resource incrementality within a fixed representation is not enough, but rather that in addition inference within a partial problem representation must be able to be interleaved with representation extension operations, so that a problem solver

can heuristically search towards an appropriate problem representation. Incrementality with respect to representation extension enables a system to reuse results from prior computations even when the representation on which those computations is based is modified between queries.

The last form of incrementality stated above may seem a bit extreme. Yet, in one recent problem solver, Wimp [5], that relied on the kind of problem solving process sketched above, the processing bottleneck was in its probabilistic representation service, precisely because the service was not incremental with respect to representation extensions.

Efficiency The goal of incrementality is efficiency. Not all efficiency concerns, however, are captured under the rubric of incrementality.

Definition 2 *A representation service is Efficient to the extent that it maximizes the information gain with respect to a query per resource increment.*

Again, we can identify several desirable forms of efficiency:

1. Efficiency with respect to network structure: There are three kinds of structure which can be exploited: the network topology, intra-distribution qualitative structure, and quantitative structure.
   (a) Network Topology: All modern belief net algorithms exploit the conditional independence information contained in the topology of a belief net to reduce computational complexity.
   (b) Intra-distribution qualitative structure: There is often considerable structure within the conditional distributions in a belief net [11], [19], [39], [17]. This structure can and should be exploited to improve efficiency. For a discussion of how this structural information is captured and exploited in SPI see [11].
   (c) Numeric structure: Finally, there is often considerable numeric structure within a belief net, in the form of skewness of distributions (a distribution is *skewed* when one of the probability masses in the distribution is larger than the others, we will formalize this later). Several systems have explored exploitation of this structure [31], [24], [22].
2. Efficiency with respect to resource incrementality: We expect an incremental system to be only minimally more expensive than a non-incremental system on comparable tasks.

3.3.2 Outline of Term Computation

A term computation approach will be interesting only if we can get a significant amount of information through the computation of a small number of terms. While there are many ways this might arise[1], we motivate the approach through the introduction of a critical assumption: we assume that most distributions in a belief net are "asymmetric:"

Definition 3 *A marginal probability distribution is* **Asymmetric** *if one mass element is larger than the remaining element(s). A conditional distribution is Asymmetric if each row satisfies this constraint. In this case it need not be the same element in each row.*

If all the distributions in a belief net are asymmetric, then most of the probability mass for many queries is contained in the first few terms[2]:

Theorem 1 *Given a Belief-net over n two-valued variables such that all distributions are asymmetric with a larger mass at least $(n-1)/n$, then the $n+1$ largest terms in the joint distribution across the variables contain a total mass of at least $2/e$.*

Note that this result is not based on any assumptions about the structure of the network. The degree of asymmetry assumed in the above theorem may seem extreme. However, it is quite natural in many applications, such as failure modeling of complex systems. Thus, our answer to the question of which terms to compute will be to compute the largest terms first. One could construct a term computation system which merely enumerated elements of the full joint distribution across all variables in a network, as in our example. Indeed, existing proposals for anytime probabilistic inference essentially do this [22], [24]. However, such an approach can be grossly inefficient. There are several sources for this inefficiency: First, there would be a time inefficiency due to unnecessary repetition of sub-computations (eg, the computation of $P(B_t|A_t) * P(A_t)$ in our example). Second, there would be space inefficiency resulting from keeping each term separate. Finally, it is not obvious how such simple methods can be made incremental with respect to newly arriving evidence, queries, or belief net extensions.

Developments in exploiting the probabilistic independence relations expressed in belief nets provide the necessary basis for designing computations which address these problems. In general, the sparser a belief net, the more finely any computation can be partitioned into independent sub-computations which share only a small number of variables. For example, given the net in figure 3.2, a query for $P(D)$ can be computed by first computing the full joint probability distribution, then marginalizing over all

[1] For example, through domain dependent knowledge of paradigmatic "cases".

[2] Proof in extended report.

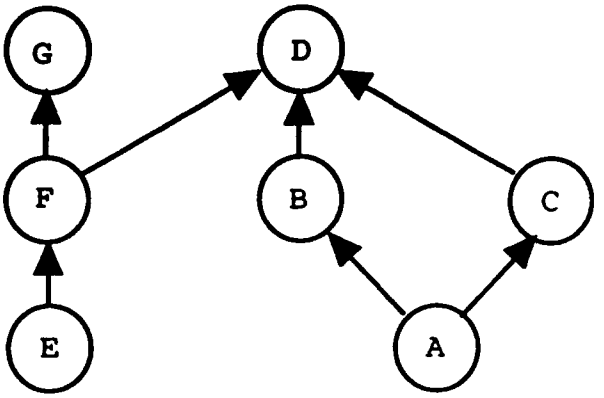


Figure 3.2: Paradigmatic Belief Net

variables except D:

$$P(D) = \sum_{A,B,C,E,F} P(D|B,C,F) * P(F|E) * P(E) * P(B|A) * P(C|A) * P(A)$$

However, a much more efficient form of the computation is:

$$P(D) = \sum_{F}(\sum_{E} P(F|E) * P(E)) * \sum_{B,C} P(D|B,C) * \sum_{A} P(B|A) * P(C|A) * P(A))$$

Having done this, we can eliminate redundant computation simply by caching intermediate results. Similarly, we can reduce the space requirement by combining terms when their bindings differ only on variables not needed in the remainder of the computation. In the extreme, each of these can reduce the corresponding complexity (time and space) for computing each term beyond the first from n to $Log(n)$, where n is the number of variables relevant to a query.

In the following section we first develop the basics of term computation (which is inherently incremental with respect to resource consumption) for a static network, set of evidence, and set of queries. We then sketch how the fundamental computation can be made incremental with respect to queries, evidence, and net extension.

The elementary primitive out of which we build a term computation system is the construction of a stream of terms for some node in the evaluation poly-tree for a set of queries. This stream will be constructed, recursively, by combining streams of terms from child nodes in the poly-tree. We first describe this evaluation poly-tree and its construction, then explain the term computation process.

Evaluation poly-tree construction Consider the net in figure 3.2, and assume our queries are for G and D. The expression for $P(D)$ has been given earlier. The expression for $P(G)$ is:

$$P(G) = \sum_F P(G|F) * (\sum_E P(F|E) * P(E))$$

It should be obvious that we can efficiently combine these two expressions into a single evaluation poly-tree:

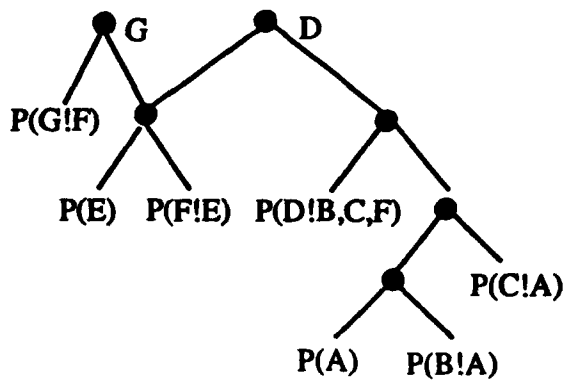


Figure 3.3: Evaluation Poly-tree for sample query set

Construction of an optimal evaluation poly-tree for an arbitrary query set is a hard problem [29]. However, simple, polynomial-time heuristics perform quite well, and are described in [14], [29]. This previous work was performed in the context of exact query evaluation (that is, computation of all terms), but the theory remains applicable, and so will not be repeated in detail here. The process used in [29] is the basis for our current implementation. That method is essentially a greedy search through the space of partial evaluation trees[3].

Term computation Given an evaluation poly-tree for a query set, the primitive operation at each node in the tree is generation of a term. Term generation is simple: each term is generated by forming the product of a term from the left child and a term from the right child. There are, however, several issues to consider: (1) Control: computations can be performed in either a data-driven or goal-driven fashion; (2) Term selection: the decision of which term to compute next; (3) Term combination: the detection of terms which can be combined. We discuss each of these issues in turn, then demonstrate computation of a term using our example poly-tree.

[3]In fact, the situation is slightly more complex when the use of local expressions is considered, since then the set of variables needed below a node in the poly-tree varies depending on the values of the variables instantiated so far. For this reason we actually interleave poly-tree construction and search.

Control Most probabilistic inference systems are data driven. SPI, the system from which the TCS is derived, however, is query driven. We retain this query driven computational style in our TCS, although it could undoubtably be adapted for data-driven computation. Computation therefore starts at one of the roots of the poly-tree when the problem solver asks for refinement of the distribution for that node, and the root queries its immediate children for terms as needed.

Term selection We earlier stated that we would attempt to minimize the number of terms computed by computing largest terms first. We use AO* to search for the largest term. AO* requires two measures, a measure of "distance so far" and a heuristic estimate of remaining distance. We use the mass computed so far as the inverted "distance traveled so far," and the partial value returned by a partial subterm as our heuristic estimate. This is an admissible heuristic, and so guarantees that the largest term will be in front of the agenda upon termination[4]. Problem solver guidance can be provided in the form of a "scaling function" which has access to term bindings and can scale the probability masses before they are used to order the search agenda.

Term combination (marginalization) The number of terms computed in response to any query is exponential in the number of relevant variables. However, the major advance offered by recent developments in probabilistic inference is reduction of the exponent for computation of complete distributions from number of relevant variables to number of relevant variables in a single factor or cluster, as we discussed earlier. We should not have to pay a higher price simply to achieve incrementality. We can achieve this efficiency by merging completed terms which are distinguished only by bindings on variables not needed at higher levels of the evaluation poly-tree. This creates two problems. First, a term which has already been incorporated into streams at higher levels in the evaluation poly-tree can suddenly have its value change (positively). Simple dependency tracking mechanisms suffice to record the information needed to update these higher terms. Second, exactly what does the AO* guarantee now mean? In poly-tree nodes where marginalization takes place, a partial term can be extended in two ways: by multiplying its value by terms from remaining distributions, or by adding additional ground terms[5]. While we use a heuristic which is admissable in its estimate of the effect of the former, our heuristic is inadmissable with regard to the latter (because it ignores marginalization). This means we can only make a relatively weak statement about terms in streams generated from poly-

[4]This selection criterion is similar to the techniques used by deKleer [16] and Henrion [22]. Both use search on restricted classes of networks for the diagnostic task of finding most likely composite hypotheses, with good results. One contribution of our work is to show how this technique can be used in a more general setting.

[5]A "ground" term is one with a unique binding for each variable in the subtree rooted by the poly-tree node under consideration.

trees containing marginalization: that the first term returned will be that term whose lower bound is highest after considering all complete ground subterms computed so far. Note that the term need not be "complete" in the sense that further ground terms may be added into it during later computation. It is, however, complete in the sense that it is a sum of a set of complete ground terms.

Complexity The key assumptions we make are that: (1) the probability distributions are sufficiently asymmetric and; (2) the graphical structure of the belief net is sufficiently sparse. Under these assumptions, the evaluation poly-tree will be such that the total number of terms computed in all streams, in the course of computing the first n term requests for each query in the query set, will be n times the number of nodes in the poly-tree. Since the poly-tree is a binary tree, this in turn is $2n$ in the number of variables relevant to the query set. All the operations we have described are either constant time, linear, or at worst $nlog(n)$ (reordering the agendas) in the number of terms in an agenda. Therefore, the total complexity, in the admittedly most optimistic case, is $2n^2log(n)$ where n is the number of variables relevant to a query set and the number of terms requested. Our experience in actually applying this procedure to three tasks, computation of marginal probabilities, most likely composite hypotheses, and complete decision analysis, confirms that this estimate is in fact realistic for a typical class of belief nets describing decision models for diagnosis and control of simple digital circuits.

Making Term Computation Incremental

The basic process sketched above is incremental with respect to computation of additional terms for a static query set. We have made this process incremental with respect to queries, evidence, and limited model reformulations (namely the extension and pruning operations needed to support the OLMA task). We omit discussion of these issues here due to lack of space. For a more complete discussion see[12].

Discussion

We have sketched a process which is essentially than heuristic search for the set of bindings across a set of variables that maximizes the posterior probability across those variables. In another context, deKleer has referred to this as the "Most Likely Composite Hypothesis" problem [15], Henrion has described an algorithm for diagnosis in very large knowledge bases [22], and Pearl has discussed the problem of "Distributed Revision of Composite Beliefs" [30]. From another perspective, Horvitz *et al* have been developing bounded conditioning as an approach to anytime probabilistic inference [24], and Boddy has proposed an anytime approach to dynamic programming [3]. We believe the contributions of our work are several: (1) We have shown how, with caching and marginalization, an incremental probabilistic inference

system based on computation of individual terms can be made as efficient at computing all terms (within a factor of *logn*) as the best algorithms for exact inference; (2) We have demonstrated that this process can be made incremental with respect to queries, evidence, and model revisions; (3) We have argued that such a system can serve as the basis for a tractable general-purpose low-level representation service. In particular, this general formulation can be used for evaluation of decision bases, as will be discussed in the next section.

3.4 Term Computation and Real-time Diagnosis

There is still a missing link in our base-level OLMA story. What does term computation have to do with evaluating a decision basis, and what does that have to do with diagnosis? Our hypothesis is that the theorem we presented for general *asymmetric* belief nets holds for typical decision bases. Further, we make a stronger hypothesis: that the computation of the few largest terms in the expected utility will be sufficient to distinguish the highest expected utility action. An expected utility is the sum of a number of terms, each of which is a probability of the system resting in a certain outcome state (given the local expression language, perhaps a partial state) multiplied by the (partial)[6] utility of that (partial) state. Our hypothesis will be valid, then, when most terms will be for either low probability outcomes or low utility aspects of an outcome. In our system, then, diagnostic reasoning is driven by the search for high payoff (either positive or negative) utility terms, which drives a search for high probability outcomes, which in turn drives a search for high probability current state information (note that, due to the factoring which takes place in the evaluation polytree construction and the structure made explicit in the local expression language, it is not obvious to state a priori what current state information will be queried during expected utility term computation.).

There is one difficulty we encountered: while we have prior probabilities to guide the instantiation of component states, we have none for action alternatives. There are seven action alternatives for the 4 gate problem (four replacement actions, two probe actions, and "waiting"), and two decision stages. We have introduced a simple heuristic to aid search: we first compute the first few most likely composite hypotheses across component states, and rate more highly those actions that replace or test components which are faulty in one or more hypotheses.

Does all this work? We have only executed the system on small digital circuits to date, due to high overhead in our prototype implementation. However, we have observed linear term computation times over a range of 1 to 16 gates. By contrast, exact evaluation of the decision basis exhausts available space (90mb swap on a Sparcstation 1+) at 8 gates. FIgure 3.4 is a typical performance graph. The task is monitoring a simple 4 gate "half-adder." The vertical axis is cost-per-failure (total scenario cost

[6] As a result of the additive value model.

divided by number of failures), and the horizontal axis is the number of cpu seconds corresponding to one "tick" of the simulation clock (thus, higher numbers simulate a faster cpu for the agent). The solid dots record the cost of running an exhaustive decision algorithm, the crosses record the cost using the TCS. These preliminary results indicate the TCS approach is considerably more robust with respect to task timeliness requirements than is exact computation.

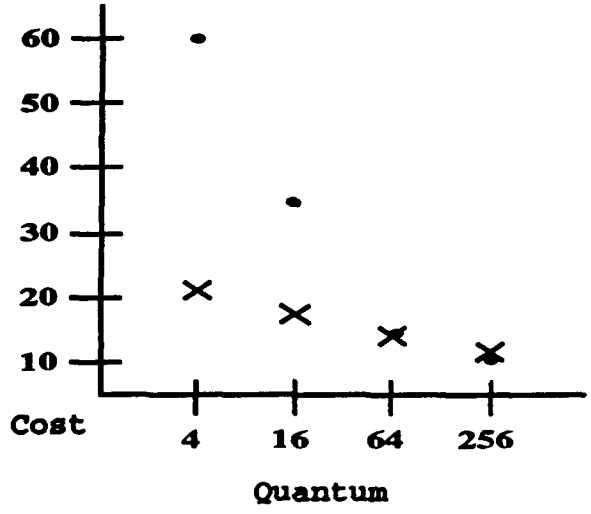


Figure 3.4: Performance of TCS vs Complete eval

Chapter 4

Reaction and deliberation in real-time diagnosis

Each TCS point in the above graph is obtained by first performing a series of runs to determine the optimal number of terms to compute for a given cpu speed, then measuring the cost using that number of terms. Thus, the number of terms computed is higher for quantum 256 than for quantum 4. This presumes that the number of terms computed on each decision cycle is constant and independent of both internal and external state. It further presumes the agent will ignore any sense data that arrives while a decision computation is in progress. This is a very naive approach to real-time problem solving. More generally, we can imagine a meta-decision process which is invoked whenever either: (1) new sense data arrives; or (2) the state of decision evaluation changes (eg, completion of the computation for an additional term). Whenever a state change occurs in either, this meta-decision process has the following options: take an action directly, initiate, terminate, or abort a decision evaluation, or push a new decision evaluation on the stack of current evaluations. Information available for this decision includes the current sense data and the current state of the computation (number of terms computed so far, most likely composite hypothesis, etc). In this chapter we explore further a formalization of the role of base and meta-level decision processes in embedded systems, and we derive a model of the role of reaction in real-time decision making from a consideration of the computational dilemma facing finite agents acting in the world. This dilemma is simply that more time spent in computation will generally provide a better solution, but more time also means more delay, which may have costs of its own. Our model provides a more fundamental role for reaction than is generally assumed. We illustrate this model with examples from our prototype OLMA, and briefly survey several current proposals for real-time problem solving architectures in light of the role they assume for reaction.

4.1 Overview

The problem of real-time decision making, or acting in time, presents a fundamental challenge to AI approaches to problem solving in general, and to decision making in particular [2], [23] . Two predominant responses to this challenge are reaction [4], [35] and meta-level reasoning [36]. We find most discussion of the roles of reaction and meta-level reasoning in real-time problem solving confusing and confused. The primary goal of this chapter is to present our understanding of the roles of and interactions among potential decision-making elements, and especially the role that reaction plays in decision making and in problem solving in general.

Our interest is in how finite agents cope with a complex and dynamic environment in the pursuit of their goals. The fundamental requirement of an autonomous agent is that it be able to *act* in pursuit of its goals. Therefore, we take the ability to choose an act, or *decide*, as primitive. We will organize our discussion of decision making around three characteristics of any decision situation: the process by which the decision will be made, and the domain of action for the decision, and the way in which a decision situations arises and is recognized.

4.2 Decision-making processes

Traditional theories of decision making or planning in AI model the process as the application of a general-purpose reasoning procedure to a problem representation stated in some language. These models generally presume the knowledge is organized in a form convenient and compact for expression, and the reasoning complexity is unbounded if the language is at least as expressive as FOPC. Such decision processes are typically termed *reflective* or *deliberative*. By the use of the term reflective we do not mean to necessarily imply any form of self reasoning, as in [41]. Rather, we simply use the term in its dictionary sense of "to think deeply." By contrast, the term *reaction* is typically used to denote a decision-making process in which the decision is made based on simple, direct sensor-effector connections, and in which either the depth of computation or the total computation time is bounded.

We find this characterization of reaction and reflection unsatisfactory. By the above definitions, processes of the complexity of finding the optimal decision given a decision basis, which are known to be NP, are "reactive:" it is straightforward to represent the computation which must be performed as an expression tree over the parameters involved, and the depth of the expression tree is linear in the number of parameters. Therefore, by the above definition, this is a "reactive" process. Yet, all of the uncertainty in a decision basis can be compiled out to yield a *decision policy*, a statement of action contingent only on hard situation observables. The fact that both forms of decision making would be classified as "reactive" by at least one common definition seems counterproductive to us. In the following we build up a model of real-time decision making processes from which we derive a very different

characterization of reaction and reflection.

4.2.1 Models of decision processes

There is an essential element missing in the above definitions: they do not take into account either the nature of the environment, the task, or the agent [13]. Given a probability distribution over domain decision situations an agent might face, and a set of utilities over outcomes, we begin by defining a *decision process*:

Definition 4 *A decision process is any computational process which recognizes a subset of situations and selects an action for each element in the subset.*

Definition 5 *An Optimal real-time decision process is a bounded-space bounded-time decision process which has the maximum expected utility over the subset of decision situtations to which it responds, where the expected utility is evaluated at the time the computation will complete when executed on the agent for which the process is designed.*

Optimal real-time decision processes for a given environment, task set, agent combination can vary in two ways: The class of situations to which they respond, and the computation time required (two decision processes which respond to the same set of situations can have indentical expected utilities even if they take differing amounts of time). Finally, we define an *optimal real-time decision-process set* as a set of decision processes that together do not exceed the resource limits of the target agent, and which as a set provide the maximum expected utility of any such set. Notice that we have not yet attempted to define reaction or reflection. The next step on the path is to consider the domain of a decision situation.

4.2.2 Decision domains

In general, decision-making can take as its domain the environment, the agent, or both. When the domain is the environment, input is the state of the environment, as evidenced by sensors, and output is an action, to be performed by effectors. When the domain is the agent (that is, meta-level decision making), input is (some aspect of) an ongoing decision process, and/or output is a modification to such a process. While it is not obvious a-priori, we conjecture that an optimal real-time decision process set for highly resource constrained agents and/or agents operating in highly dynamic environments will often include meta-level decision processes (for example, meta-level reasoning would permit a highly resource constrained agent to have only a subset of its optimal real-time decision process set active at any one time, reducing overall resource requirements for a given set of base-level decision processes).

Meta-level decision making requires modeling ongoing decision processes. Any computational process can be modeled as consisting of three aspects, data state,

control state, and procedure. At this level of abstraction we can say little more. Data state is simply the declarative input/output interface to a process, and its form is determined by the language the decision-making process requires. Similarly, control state cannot be specified separately from procedure. We chose to commit to decision theory [37] as our theoretical model of decision making, and the *decision basis* [25] as our representation of a decision problem. This defines a data state as consisting of five sets:

- Parameters representing aspects of the state of the domain,
- Beliefs about the values of these parameters and their interrelationships,
- Action alternatives,
- Beliefs about the effects of actions on parameters, and
- Preferences over possible outcomes.

A complete specification of control state would be impractical and extremely implementation specific. However, our inference method suggests a natural abstraction: the number of terms computed. Similarly, our method has a single procedural choice-point: whether to use AO* or best-first search (this could be considered control state, but we choose to represent it as an alternative procedure for pedagogy).

4.3 Decision Situations

We are almost ready to define a reactive decision process. One question remains: how does a decision situation arise? We assume a symbolic, event-based interface to the external world, which makes the first question simple for base-level decision situations: a new decision situation arises whenever new symbolic data arrives from the world. At the meta level the same principle applies: a new decision situation arises whenever the state of the active decision process or the external world changes. Of course, simply because a decision situation arises does not mean that there must be a decision process which responds to it.

Now we face a second problem: how is an agent to recognize that a new decision situation has arisen? All decision processes could be active simultaneously, monitoring for situations to which they can respond. However, this approach is in general infeasible: for arbitrary forms of meta-level reasoning the number of potential decision processes could be unbounded. An alternative is to seek the other extreme: to find a minimal set of decision processes which must be active at all times. This minimal decision process set must be sufficient to generate all possible decision processes in the full set under appropriate circumstances, and will in general contain three kinds of decision process:

1. decision processes that yield an action directly.

2. meta-level decision processes that initiate new decision processes.

3. meta-level decision processes that modify existing decision processes.

While not required by our model, the easiest way to ensure that this set is closed and finite is to require that all decision processes in our minimal set be stateless (and therefore not the objects of meta-level decision processes). This permits them all to be executed by a computational element which need not monitor its own state. We are finally in a position to define a reaction:

Definition 6 *A reaction is a stateless, always active decision process, and the set of such processes necessary to initiate all decision processes in a larger decision process set is the minimal reaction set for the larger decision process set.*

It may be that there exist stateless decision processes not in this set which, if added to the set, would improve the expected performance of the agent (by reducing response time to the decision situation responded to by those processes). The minimal reaction set, supplemented by the subset of such stateless decision processes which maximizes expected performance of the agent, is termed the *optimal real-time* reaction set. For convenience, we term all decision processes not in the reaction set *reflective.*

4.3.1 Real-Time Decision Making

We can now present a general computational model of real-time decision making. First, we posit a decision maker consisting of at least two computational elements, a reactive processor and a reflective processor. The reaction set will be located in the reactive processor. Since all decision processes in the reaction set are stateless, the reactive processor can be implemented as a combinational circuit. Input to the reactive processor must include all state change information, both external to the agent and internal. The output of the reactive processor can be any one of the following:

1. An action to be performed in the external world. In this case the reactive element is responding to an external decision situation by choosing to make the decision reactively, and producing the decision.

2. A change to an ongoing reflection. In this case the reactive element is responding to a meta-level decision situation by choosing to make the decision reactively, and again producing the decision.

3. A new reflection to be begun. In this case the reactive element is responding to a base or meta-level decision situation by choosing to make the decision reflectively, and initiating the reflection.

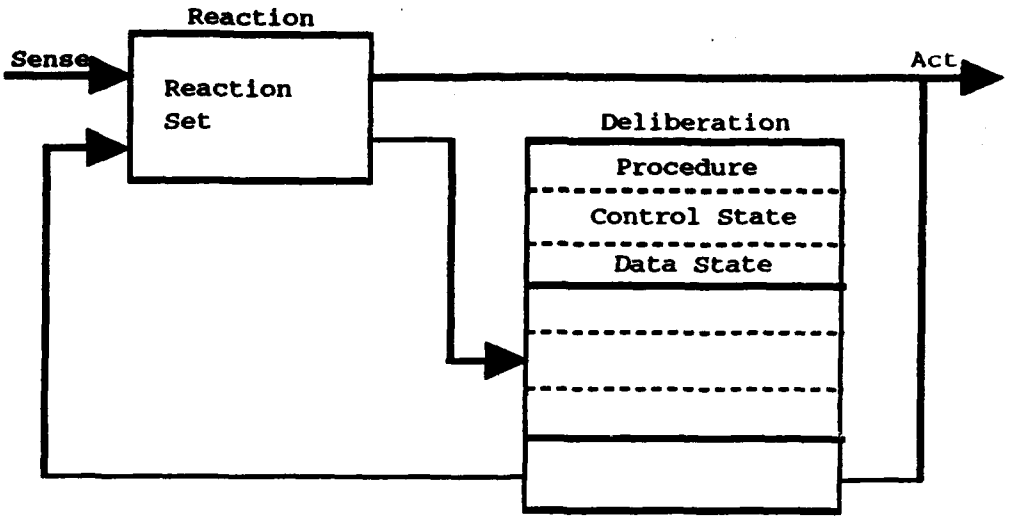


Figure 4.1: Reaction and Reflection

Of course, the reactive element can always chose to ignore a decision situation entirely. Note that in this model all reflection is initiated by reaction. When initiating a reflection, the reactive element must provide initial values for the data state, control state, and procedure. If the data state scope includes only the external environment, base level agent action, and a problem statement, then the reflection spawned is base-level. However, if the scope includes one or more elements of other on-going reflections (data, control, or procedure), then by definition the new reflection is meta-level. We assume that only the highest level reflection currently outstanding is active, and that all lower level reflections are suspended until it completes or is terminated by the reactive element. A diagram of the necessary communication and control relations among the reactive and reflective elements is shown in figure 4.1. Variables in reactive decision processes are bound to the currently active decision process in the reflective element.

We have introduced a variety of definitions, and claimed to provide a general model of real-time decision making in terms of those definitions. Clearly these definitions are useful only to the extent that we can do something with them. Ideally, we would like to provide a procedure for solving the design problem implicit in the model: given an ETA description, derive the optimal real-time decision process set and corresponding optimal real-time reaction set. We have not yet produced such a result, and doubt it is achievable in the near future (if ever) for non-trivial problems. Rather, we present two illustrations of the utility of this framework. First we illustrate this model with some examples from an agent we are constructing. Second, we survey, from the perspective of our model, several current proposals for real-time problem solving architectures.

4.4 Reaction and Deliberation in the OLMA

As a consequence of our model of the roles of reaction and reflection, we can fully specify an agent by specifying its reaction set. However, reactions that initiate reflections must specify the input over which the reflection is to operate, the initial control state, and the procedure to be used. Initial data state includes prior beliefs about equipment state, current sense data, and, as it is computed during the course of inference, the most likely current hypothesis on equipment state. Control state is the number of terms computed and will always be initialized to zero. Finall, we will consider the best-first search form of term computation as our only procedural choice.

Sample Reactions for the OLMA We present below a few sample reactions from an OLMA we are constructing to maintain simple digital circuit: a half-adder. First we present the minimal reaction set needed to collect the data presented earlier on performance of term computation.

Reaction 1: Start a deliberation

Event: New sense data from the world.
State: No reflection currently in progress.
Action: Initiate deliberation on base-level ID.

Reaction 2: Terminate reflection.

Event: New term computed, #terms > limit.
State: Base level reflection currently in progress.
Action: Terminate base-level decision process and take recommeded action.

Next we look at a slightly more sophisticated reaction. As the sensing frequency increases, it becomes worthwhile to monitor data state during deliberation. If the most likely composite hypothesis is that the equipment is OK, then in pays to abandon the current inference, and prepare to process the next available sense data:

Reaction 3: Reactively terminate base level inference if MLCH is OK.

Event Reflective state change: new possible world state identified.
State: MLCH = OK
Action: Abort base-level decision process.

4.5 Discussion

It may seem that the distinction between the base-level decision algorithm and the reactive/reflective meta-level procedures is arbitrary: couldn't all this be collapsed into one base-level procedure (eg, include in the base level decision procedure a heuristic which evaluates whether or not to stop on each iteration)? In this particular case we could in fact collapse the set of meta-level processes into the base level process,

except for the processes that monitor for new world state. Since world state changes concurrently with agent state, tracking world state while executing a decision process requires concurrent decision processes.

We have developed a model of the role of reaction in real-time decision making, but it is lacking in several important respects. First, we have provided no guidance on how to find an optimal reaction set, given an ETA specification, or even how to evaluate any given reaction set. We have explored the problem of building optimal reaction sets, with basically negative results so far. Base-level reactions seem simple on first examination. However, there are several problems: first, the optimal agent is NOT stateless, and any attempt to build a stateless reaction set is doomed to failure. State, however, introduces its own complexities: how much, how is it to be maintained, and how large will the reaction set grow? A clean theoretical formulation of state is simply the sense/act history of the agent. This, however, is clearly impractical, as it grows linearly with time. An alternate formulation, which we are in the process of exploring, builds reactions for those situation/action pairs which can be detected with finite state.

Definition 7 *$R_n(S_n, A)$ is a relation from sense sequences of length n (including acts up to $n-1$) to actions. $r_n(s_n, a)$ is one member of the relation.*

This relation will not, in general, cover all of S_n, but only those elements for which the action is appropriate regardless of prior state. We can detect members of R_0 through R_n on-line as follows:

1. Keep the full decision model for n past stages.
2. When a decision is made, test its sensitivity to the believed equipment state for time $t-1$, $t-2$, etc, where t is the current time.
3. If the decision is insensitive to equipment state at time $t-i$, then the sense/act history for $t-i$ through t is a member of R_{i+1}.

The key step is the determination of independence from prior state. We are developing symbolic methods for establishing this independence, given the sense history.

Meta-level reactions pose an even more serious problem, for the following reasons:

1. Current theories for controlling base level inference [23], [36] do not consider repetitive decision making of the kind the OLMA faces, and so are inapplicable.
2. A model of the meta-level decision problem which has the markov property is intractably large, and so reinforcement learning schemes are inapplicable.

We regard this as a major open problem in real-time problem solving, and expect it to be a focus of future research.

Another limiation of the work described here is that we have only illustrated this mechanism with bounded-time decision processes over a decision basis, a simple propositional language. We do not believe such a simple language provides an adequate account for the background knowledge an agent can bring to bear on a decision situation. All these are topics for future research. Another issue is how reactions evolve. Conceptually this is not a problem: we did not say that reactions could not be reflected over, but only that they were not visible to the reactive element and therefore could not be modified during the course of a single execution.

Nonetheless, we believe the contribution made by our model as developed to date is significant. A wide variety of "architectures" for autonomous agents have been proposed, and there has been no way in general of comparing these architectures on any dimension. Using the model of reaction presented in this paper, we can identify clear differences in the role of reactive processes among various architectures. Note we do not claim that an implementation that provides all modes of interaction is necessarily better than one which does not: that is an ETA specific question we do not claim to have the answer to. Nonetheless, we examine a few proposed architectures in the paragraphs that follow.

One standard role for "reaction" is as a front end to a standard AI problem solver [6]. Architectures of this type typically allow the "higher," "deliberative" levels to control the reactive level. An important capability of our reactive model is missing in this approach: the ability for the system to monitor the changing state of the world, and coordinate the state of reflective problem solving to that of the environment. Also, such systems must provide a separate mechanism for controlling meta-level reasoning, if they provide any account of it at all.

A second significant problem solving model is that of SOAR [26]. The SOAR model seems very close to ours, with one exception: decision situations in SOAR arise, not on any state change, but only on one specific kind: problem-solver blockage. Thus, SOAR loses the capability for opportunistic meta-level control, either reactively or reflectively. SOAR can simulate this capability by blocking on every state change, but this would impose a meta-level cycle between each base-level step, an exorbitant overhead.

BB* [18] is a blackboard system which interposes a single level of meta-level reasoning between each base-level cycle. In recent work a front-end data processor has been added to handle data reduction tasks. Primary control resides in the reflective processor, the blackboard. This makes flexible multi-level meta-level reasoning impossible: in order to prevent infinite regress, ultimate control must reside in reaction. In BB* we can consider both the front end data processor and the meta-level reasoning as parts of the reactive system. Viewed this way, the reactive system has reasonable control capability: through introduction of data and control of the agenda reaction can control both the data and control state of reflection. However, the blackboard implemented meta-level is not necessarily a bounded time computation, arbitrary depth of meta-level computation is impossible, and the inherent overhead is high enough

that we suspect the optimal reaction set will be small, limiting the performance of the optimal real-time decision process set.

Finally, we consider the IRMA model of [34]. In this model an agent has a set of intentions (partial plans) which is constantly being merged with new opportunities identified by monitoring the world. Viewed from our perspective on reaction and reflection, IRMA has a reflective planner (the visible state is the intention set), with a reactive element that modifies only the control state of the planner, and which never initiates meta-level reflections.

4.6 Summary

Reaction is typically assigned the role of serving as a "quick and dirty" front end to a more "intelligent" deliberative (reflective) process. We have argued that reaction plays a much more fundamental role in real-time decision making: that it is the ultimate ground of all reflective processing, and that the essence of a reaction is that it is a stateless decision process which is always active. We have introduced the notion of an optimal real-time reaction set as a complete specification of the optimal solution to a real-time decision problem, where the problem specifications include environment, task, and agent characteristics. Finally, we have found that our model of reaction is adequate to compactly describe the kinds of decision making dynamics we have found useful in a study domain, the on-line maintenance agent.

Chapter 5

Summary

Monitoring, assessment, and diagnosis are often thought of as isolated tasks in theoretical reasoning. We have presented a decision-theoretic interpretation of these as tasks in practical reasoning and sketched components of our approach to this task. These include an embedded agency, decision-theoretic formulation of the task, a tractable incremental decision procedure, and a real-time architecture that provides, a new, fundamental role for "reaction" in computational architectures. The approach makes significant strides to integrating commonly held logical and probabilistic models of diagnosis, as well as incorporating real-time embedded agency concerns.

We are aware of several limitations in our current approach. Our model of time is quite naive and limited. We have no tractable methods for discovering optimal policies for control of inference. Our current inability to handle continuous variables is a serious restriction. But perhaps the most severe limitation is the assumption of a complete base-level decision model applicable to all situations. We have concurrent work which has made considerable progress on the problem of continuous variables, and are exploring methods by which situated decision models can be dynamically constructed from background domain theories. This latter work uses a meta-level version of the same base level architecture sketched here, and uses both bottom up and top-down construction methods. This, we believe, is the most exciting direction in which our research is focused.

Chapter 6

Grant Administration Data

6.1 Publication

An article describing our approach to real-time is being submitted to is in preparation, for submission to either AI Journal or the new journal on Real-time Computing. Papers also appearred at DX-92 and SOAR-92, and are under preparation for submission to UAI-93 and CAIA-94.

6.2 Professionals involved

- Bruce D'Ambrosio, Associate Professor, OSU.
- Thomas Dietterich, Associate Professor, OSU.
- Lothar Kaul, MS, OSU, Spring, 1991.
- Caryl Westerberg, MS expected Spring, 1993.

6.3 Interactions

1. Presentation, IRTPS kickoff at WPAFL, Spring, 1991.
2. Presentation, Uncertainty workshop at Canadian AI conf, Vancouver, CA, Spring, 1992.
3. Presentation, AAAI-92 workshop on tractable reasoning.
4. Presentation, SOAR-92 (Space Operations and Research-92).
5. Presentation, Principles of Diagnosis 92.

6. Invited, Approximate Computation workshop, Real-time systems conference, winter, 1992

7. Invited Participant, NSF Workshop on Real-time and AI, Spring, 1993.

Bibliography

[1] M. Agosta. Conditional inter-causally interdependent node distributions ... In *Proceedings of the Seventh Annual Conference on Uncertainty in Artificial Intelligence*, pages 9–16. Morgan Kaufmann, Publishers, July 1991.

[2] P. Agre. *The Dynamic Structure of Everyday Life*. PhD thesis, MIT, 1989.

[3] M. Boddy. Anytime problem solving using dynamic programming. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 738–743. AAAI, July 1991.

[4] R. Brooks. A layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[5] E. Charniak and R. Goldman. A probabilistic model of plan recognition. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 160–165, 1991.

[6] P. Cohen and *et al.* Trial by fire: Understanding the design requirements for agents in complex environments. *AI Magazine*, 10(3):32–48, 1989.

[7] G. Cooper. A method for using belief networks as influence diagrams. In *Proceedings of the 1988 Workshop on Uncertainty in AI*, pages 55–63. Association for Uncertainty in AI, August 1988.

[8] B. D'Ambrosio. Process, structure, and modularity in reasoning under uncertainty. In *Proceedings of the 1988 Workshop on Uncertainty in AI*, pages 64–72. AAAI, August 1988.

[9] B. D'Ambrosio. Symbolic probabilistic inference. Technical report, CS Dept., Oregon State University, 1989.

[10] B. D'Ambrosio. Incremental evaluation and construction of defeasible probabilistic models. *International Journal of Approximate Reasoning*, July 1990.

[11] B. D'Ambrosio. Local expression languages for probabilistic dependence. In *Proceedings of the Seventh Annual Conference on Uncertainty in Artificial Intelligence*, pages 95–102, Palo Alto, July 1991. Morgan Kaufmann, Publishers.

[12] B. D'Ambrosio. Term computation systems. Technical report, CS dept., Oregon State University, 1991.

[13] B. D'Ambrosio and M. Fehling. Constrained rational agency. In *Proceedings of the International Conference on Systems, Man, and Cybernetics*. IEEE, November 1990.

[14] B. D'Ambrosio and R. Shachter. Symbolic probabilistic inference. Technical report, Oregon State Univ. CS dept., 1992.

[15] J. de Kleer. Focusing on probable diagnoses. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 842–848. AAAI, July 1991.

[16] J. de Kleer and B. Williams. Diagnosis with behavioral modes. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1324–1330. IJCAI, August 1984.

[17] D. Geiger and D. Heckerman. Advances in probabilistic reasoning. In *Proceedings of the Seventh Annual Conference on Uncertainty in Artificial Intelligence*, pages 118–126. Morgan Kaufmann, Publishers, July 1991.

[18] Barbara. Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26(3):251–323, July 1985.

[19] D. Heckerman. A tractable inference algorithm for diagnosing multiple diseases. In *Proceedings of the Fifth Conference on Uncertainty in AI*, pages 174–181. August 1989.

[20] D. Heckerman, J. Breese, and E. Horvitz. The compilation of decision models. In *Proceedings of the Fifth Conference on Uncertainty in AI*, pages 162–173, August 1989.

[21] M. Henrion. Towards efficient probabilistic diagnosis with a very large knowledge-base. In *AAAI Workshop on the Principles of Diagnosis*, 1990.

[22] M. Henrion. Search-based methods to bound diagnostic probabilities in very large belief nets. In *Proceedings of the Seventh Annual Conference on Uncertainty in Artificial Intelligence*, pages 142–150. Morgan Kaufmann, Publishers, July 1991.

[23] E. Horvitz, G. Cooper, and D. Heckerman. Reflection and action under scarce resources: Theoretical principles and empirical study. In *Proceedings of IJCAI89*. IJCAI, August 1989.

[24] E. Horvitz, H. J. Suermondt, and G. Cooper. Bounded conditioning: Flexible inference for decisions under scarce resources. In *Proceedings of the Fifth Conference on Uncertainty in AI*, August 1989.

[25] R. Howard and J. Matheson. Influence diagrams. In Howard and Matheson, editors, *The Principles and Applications of Decision Analysis.* Addison-Wesley Publishing, Menlo Park, Calif., 1984.

[26] J. Laird, A. Newell, and P. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64, 1987.

[27] S. Lauritzen and D. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, B 50, 1988.

[28] Z. Li. Experimental characterization of several algorithms for inference in belief nets. Technical report, Master's thesis, CS Dept., Oregon State University, 1990.

[29] Z. Li and B. D'Ambrosio. An efficient approach to probabilistic inference in belief nets. In *Proceedings of the Annual Canadian Artificial Intelligence Conference*. Canadian Association for Artificial Intelligence, May 1992.

[30] J. Pearl. Distributed revision of composite beliefs. *Artificial Intelligence*, 33(2):173–216, 1987.

[31] J. Pearl. Evidential reasoning using stochastic simulation of causal models. *Artificial Intelligence*, 32(2):245–258, 1987.

[32] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, Palo Alto, 1988.

[33] Y. Peng and J. Reggia. A probabilistic causal model for diagnostic problem solving - part 1: Integrating symbolic causal inference with numeric probabilistic inference. *IEEE Trans. on Systems, Man, and Cybernetics: special issue on diagnosis*, SMC-17(2):146–162, 1987.

[34] M. Pollack and M. Ringuette. Introducing the tileworld: Experimentally evaluating agent architectures. In *Proceedings Eighth National Conference on AI.* pages 183–189. AAAI, August 1990.

[35] S. Rosenschein. Synthesizing information tracking automata from environment descriptions. In *Proc. First Intl Conf on Principles of Knowledge Representation and Metareasoning*, 1989.

[36] S. Russell. Principles of metareasoning. In *First International Conference on Principles of Knowledge Representation and Reasoning.* AAAI, 1989.

[37] L. Savage. *The Foundations of Statistics.* Dover Publications, 1972.

[38] R. Shachter, B. D'Ambrosio, and B. DelFavero. Symbolic probabilistic inference in belief networks. In *Proceedings Eighth National Conference on AI*, pages 126–131. AAAI, August 1990.

[39] R. Shachter and R. Fung. Contingent influence diagrams. Tech report, Dept. of Engineering Economic Systems, Stanford University, September 1990. In preparation.

[40] H. Simon. *Sciences of the Artificial.* MIT Press, 1974.

[41] B. Smith. *Reflection and Semantics in a Procedural Language.* PhD thesis, MIT, 1984.